View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic Archive of Kyiv Polytechnic Institute

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ О.В. Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Розробка програмного додатку «Навчання» для системи підготовки науково — педагогічних кадрів вищої кваліфікації в аспірантурі університету»

Виконав:
студент IV курсу, групи ТМ-61 _____
Слюсарчин Сергій Віталійович

Керівник:
професор, д. т. н.
Гаврилко Євген Володимирович _____

Рецензент:
Професор, д. т. н.
Барабаш Олег Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент Слюсарчин Сергій Віталійович

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр Коваль
(підпис)
”_ __”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Слюсарчину Сергію Віталійовичу_____
(прізвище, ім’я, по батькові)

1.Тема роботи _____
Розробка програмного додатку “Навчання” для системи підготовки науково – педагогічних кадрів вищої кваліфікації в аспірантурі університету_____
керівник роботи Гаврилко Євген Володимирович, д. т. н., професор_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи
мова програмування JavaScript, програмна платформа Node.js, середовище розробки JetBrains WebStorm_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
проаналізувати проблему навчання спеціалістів науково–педагогічних кадрів вищої кваліфікації, розробити систему для підготовки науково-педагогічних кадрів. _____
_____

5. Перелік ілюстративного матеріалу
схеми архітектури програмного продукту, знімки інтерфейсу, знімки структури проекту_____

7. Дата видачі завдання ”04.01”___Січня_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	05.01.2020	
2.	Вивчення та аналіз задачі	10.02.2020	
3.	Розробка архітектури та загальної структури системи	20.03.2020	
4.	Розробка структур окремих підсистем	21.04.2020	
5.	Програмна реалізація системи	25.05.2020	
6.	Оформлення пояснювальної записки	30.05.2020	
7.	Захист програмного продукту	06.06.2020	
8.	Передзахист	09.06.2020	
9.	Захист	19.06.2020	

Студент _____Слюсарчин С.В._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ __Гаврилко Є.В._____
(підпис) (прізвище та ініціали,)

Реферат

Актуальність теми. Всі попередні роки, була така проблема, як якісний та легкий процес навчання у вищих навчальних закладах. Тому було розроблено систему, яка б спростила даний процес, а саме, якщо розглядати сторону аспіранта то це подачу адокументів, перегляд змін в рейтингових списках, перегляд навчального розкладу, індивідуального плану, особистого профілю та занять. А для викладча, можливість швидко та зручно оцінити аспірантів, внести облік відвіддуваня пар та екзаменів, також перегляд особистого кабінету у якому знаходиться коротка особиста інформація. Загалом система має на меті економію ресурсів, та пришвидшення усіх процесів навчання.

Мета дослідження. Полягає в розробці нових підходів для проведення зручного процесу навчання вищих навчальних закладів на третьому освітньому рівні. Та створення SPA для аналізу навчанні спеціалістів науково-педагогічних кадрів.

Для досягнення поставленої задачі були сформульовані наступні **завдання дослідження**, що визначили логіку дослідження та його структуру:

— зробити аналіз всіх існуючих програмних рішень;

— проаналізувати від початку до кінця процес навчання;

— після аналізу визначити які завдання потрібно реалізувати у програмному продукті;

— оптимізувати всі існуючі процеси для роботи із системою навчання;

— розробка програмного продукту “Навчання”.

Об’єктом дослідження є комп’ютерні технології та інформаційні системи.

Предметом дослідження є комп’ютерні технології в області навчання науково-педагогічного вищої кваліфікації в аспірантурі університету.

Практичне значення отриманого результату. Практичним значенням дипломної роботи полягає в розробці системи “Навчання” та супроводження науково – педагогічних кадрів за допомогою даної системи

ABSTRACT

Actuality of theme. All previous years, there was such a problem as a quality and easy learning process in higher education. Therefore, a system was developed that would simplify this process, namely, if we consider the side of the graduate student, it is the submission of documents, review of changes in ranking lists, review of the curriculum, individual plan, personal profile and classes. And for the teacher, the opportunity to quickly and easily evaluate graduate students, keep records of vapors and exams, as well as view the personal account in which there is a brief personal information. In general, the system aims to save resources and speed up all learning processes.

The aim of the study. Is to develop new approaches to a convenient process of higher education at the third educational level. And the creation of SPA for the analysis of training of specialists of scientific and pedagogical staff.

To achieve this goal, the following research objectives were formulated, which determined the logic of the study and its structure:

- to analyze all existing software solutions;
- analyze the entry process from beginning to end;
- optimize all existing processes for working with the training system;
- development of the software product “Training”.

The object of research is computer technology and information systems.

The subject of the research is computer technologies in the field of training of scientific and pedagogical staff of the highest qualification in the postgraduate course of the university**.**

The practical significance of the result. The practical significance of the thesis is to develop a system of "Training" and support of scientific and pedagogical staff with this

system

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система.

JavaScript – мова програмування.

HTML – мова розмітки WEB-сторінки.

CSS – каскадна таблиця стилів.

JetBrains WebStorm - середовище розробки IDE.

SPA – Single Page Application–односторінковий додаток.

Фреймворк – програмна платформа, визначающа структуру програмної системи.

Маршрутизація – визначення маршрутів додатку.

ВСТУП

Для вирішення проблеми у навчанні науково-педагогічних кадрів вищої кваліфікації було прийняте рішення розробити систему у якій, було б зручно аспірантам та викладачам спростити процес навчання.

Дана система була призначена для спрощення навчання науково-педагогічних кадрів вищої кваліфікації в аспірантуру університету. Також система була розроблена так, щоб вона складалася із трьох частин, аспіранта, викладача та адміністратора.

Перша частина системи , яка розроблена для аспіранта дозволяє їм переглянути основну інформацію про себе у своєму особистому кабінеті, також індивідуальний план, знаяття та розклад і все це за допомогою декількох натискань клавіш мишки.

Дурга частина системи розроблена для викладача, де він має змогу оцінити аспіранта та внести дані про відвідування придметів.

Третя частина системи дозволяє керувати процесом навчання аспірантів, а саме, переглядати інформацію про аспірната, вносити дані про відвідування та здійснювати оцінювання.

Така система дає можливість спростити наявний процес навчання, автоматизувати роботу, яка виконувалася працівниками.

У першому розділі пояснювальна записка розгортає мету і опис поставленої задачі.

У другому розділі розробка проекту, створення програмного продукту.

У третьому розділі було проведено огляд всіх методів реалізації програмної системи.

У четвертому розділі описується вся програмна реалізація.

У п’ятому подана інструкція користувача для роботи із системою.

1. ЗАДАЧА РОЗРОБКИ ПРОГРАМНОГО ДОДАТКУ “НАВЧАННЯ” ДЛЯ СИСТЕМИ ПІДГОТОВКИ НАУКОВО-ПЕДАГОГІЧНИХ КАДРІВ ВИЩОЇ КВАЛІФІКАЦІЇ В АСПІРАНТУРУ УНІВЕРСИТЕТУ

Метою розробки була здійснена реалізація програмного продукту за структурою WEB-додатка, що в свою чергу надає таку можливість, як спрощення процесу навчання для науково-педагогінчинх кадрів вищої кваліфікації в аспірантурі університету.

Призначення даного додатку “Навчання” в тому, щоб надати користувачу чіткого і зрозумілого UX інтерфесу. При розробці продукту були розв’язані такі завдання:

1. проведено аналізи схожих по роботі систем на ринку;
2. створено тех. Завдання для системи;
3. була змодельована структура системи;
4. розроблено програмний додаток;
5. одразу ж відбулося тестування розробленого додатку на функціональність.

Потрібні можливості, які буде забезпечувати система, є:

1. можливість відмічати відвідування аспіратна.
2. можливість аспірантом переглядати особисту інформацію, розклад, заняття, оцінки та індивідуальний план.
3. можливість оцінювання викладачами аспірантів.
4. можливість редагування усіх даних про навчання.
5. кросплатформність.

1.1 Характеристика програмної системи

Основні функції програмної системи з боку аспірнта є:

1. функціонал подачі реєстраційної заяви для участі у вступній компанії після якої він має змогу навчатися .
2. перегляд особистої інформації.
3. функціонал перегляду індивідуального плану;
4. функціонал перегляду занять.

Основні функції програмної системи з боку викладача є:

1. перегляд особистого кабінету у якому розміщенна основна інформація про викладача.
2. перегляд занять.
3. можливість оцінювати аспіранта та відмічати відвідування занять.

Основні функції програмної системи з боку адміністратора є:

1. функціонал стоврення груп.
2. функціонал розприділення по групам.
3. функціонал створення новго викладача.
4. функціонал прийому на навчання.

Сторінка реєстраційної форми містить багато полів для заповнення, а саме:

1. прізвище;
2. ім'я;
3. по батькові;
4. стать;
5. дата народження;
6. спеціальність;
7. факультет;
8. кафедра;
9. форма навчання;
10. форма оплати;
11. іноземна мова вступника;
12. ВНЗ, який було закінчено;
13. рік випуску з ВНЗ;
14. освітній ступінь;
15. освітній ступінь з відзнакою;
16. електронна пошта;
17. номер телефону;
18. потрібність гуртожитку на час навчання;
19. середній бал диплому із 0 – 100;
20. кількість публікацій;
21. прізвище наукового керівника;
22. особливі відзнаки;
23. Додаткова інформація;

Після завершення всіх пунктів реєстрації, дані аспіранта заносяться до БД і адміністратор системи може обробити дані реєстраційної форми.

Зовнішній інтерфейс адміністратора складається із сторінки входу та сторінки для роботи із даними аспірантів.

На рисунку 1.1 зображена діаграма прецедентів системи.

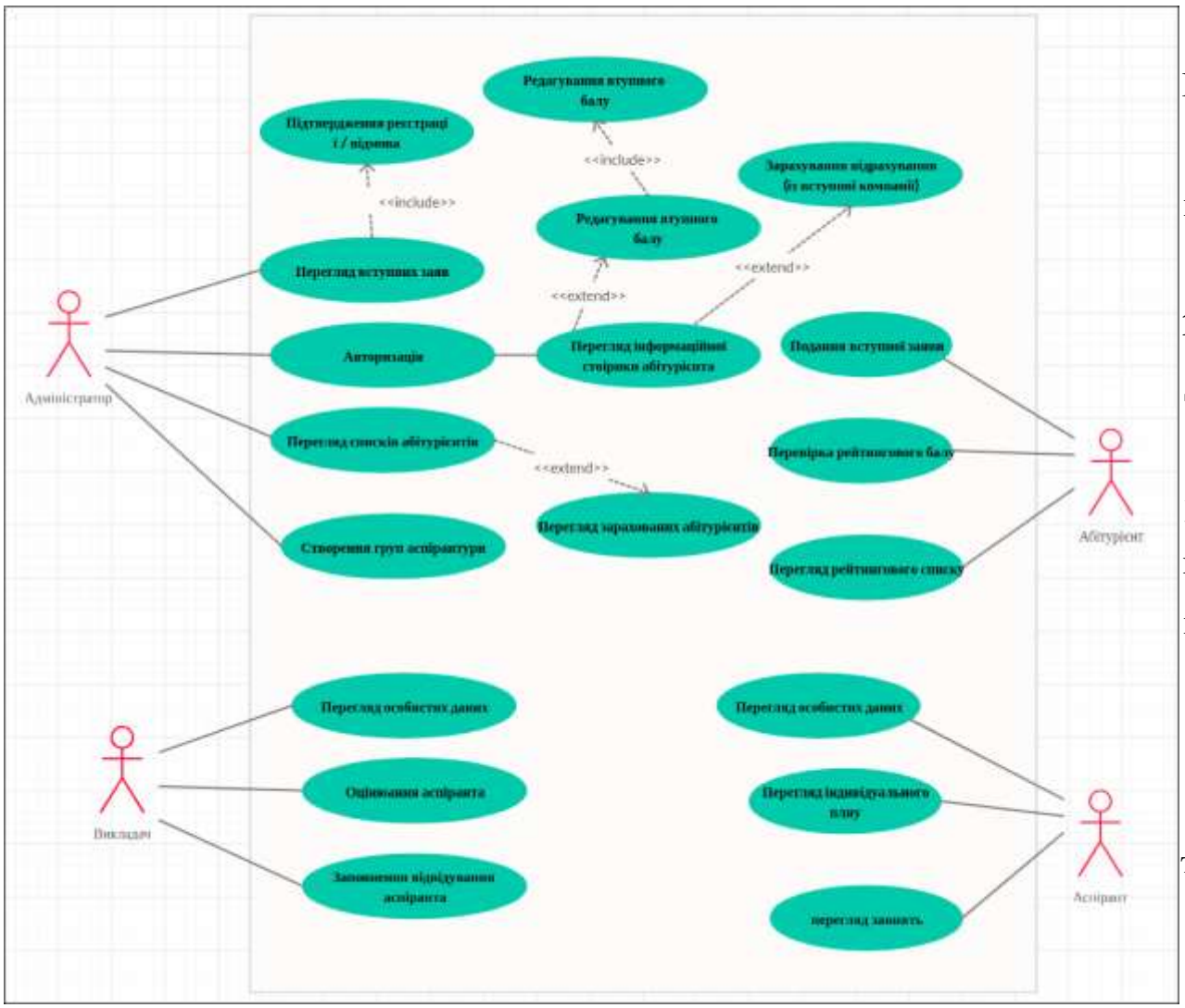


Рисунок 1.1 Діаграма прецедентів системи

На діаграмі послідовності системи можемо побачити, що для ролі вступника (Рисунок 1.2) показана взаємодія користувача із системними елементами. Абітурієнт має можливість ввести свої особисті дані до реєстраційної форми. Заява після реєстрації автоматично зберігається у базі даних, також система показує сповіщення, що користувач успішно зареєстрований.

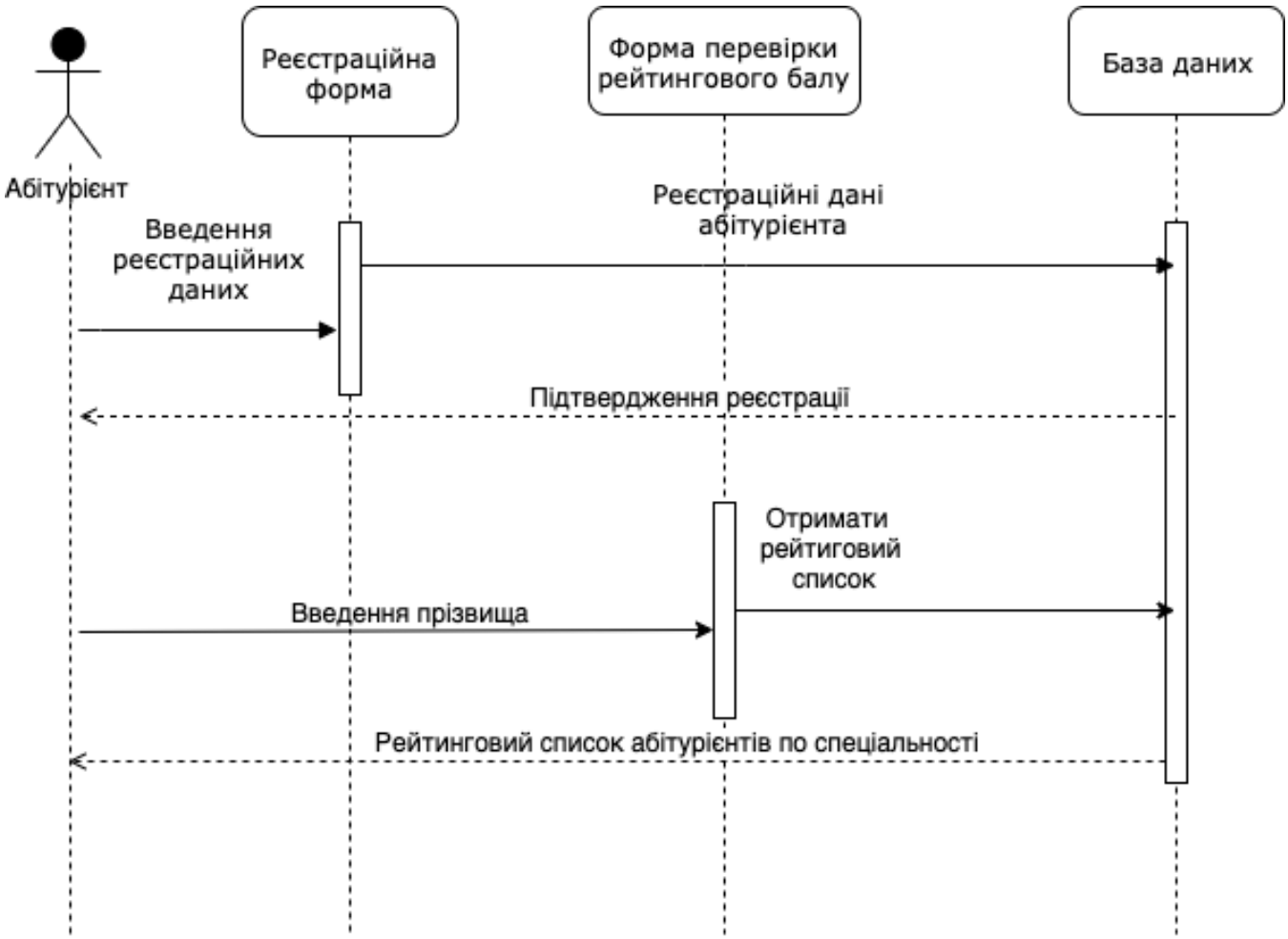


Рисунок 1.2 Діаграма послідовності системи із правами абітурієнта

На діаграмі послідовновності програмної системи (рисунок 1.3) схематично зображено взаємодію користувача з елементами системи. Адміністратор має змогу пройти авторизацію в програмній системі за допомогою пари логін-пароль. В разі успішного входу адміністратор може переглядати список заяв вступників, підтверджувати або відміняти їх реєстрацію, переглянути список підтверджених абітурієнтів, вносити бали за екзамени та робити зміни у наявних даних абітурієнтів. Всі зміни автоматично враховуються та відображаються у рейтинговому списку.

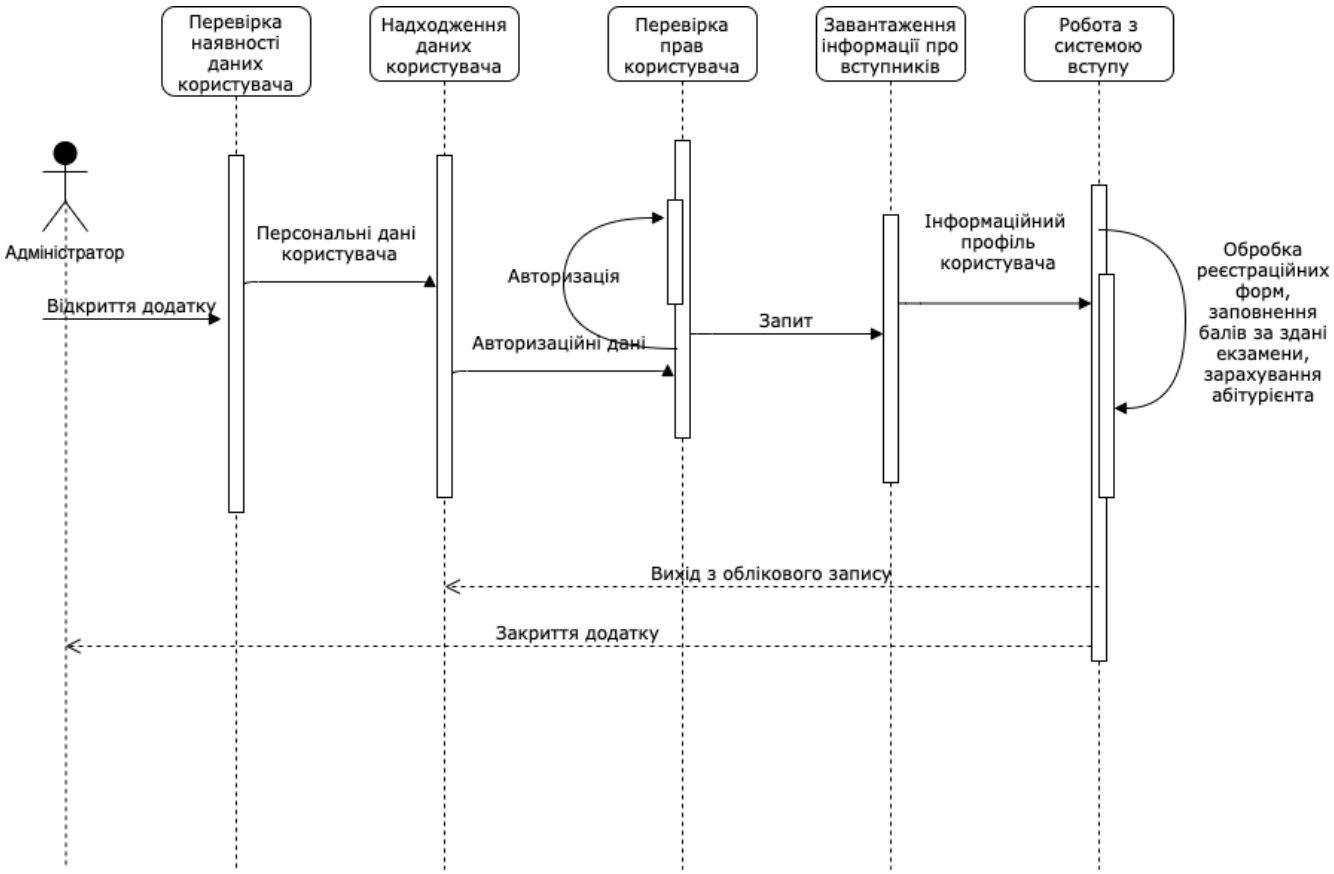


Рисунок 1.3 Діаграма послідовності системи із правами адміністратора

Висновки до розділу 1.2

У цьому розділі показні вирішення проблеми підготовки науково – педагогічних кадрів вищої кваліфікації в аспірантурі університету.

Система розділяється на три підсистеми:

1. підсистема адміністратора;
2. підсистеми викладача.
3. підсистема абітурієнта

У першій адміністратор приймає певний ряд рішень із зарахування на навчання за допомогою перегляду та редагування даних абітурієнта.

У другій підсистемі із правами викладача, можна переглядати свою особисту інформацію та здійснювати оцінювання та відмічати відвідування занять.

У третій підсистемі із правами адміністратора, можна подати заяву і переглянути рейтингові списки.

2.ОГЛЯД МЕТОДІВ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Під час розробки системи був зроблений правильний вибір програної реалізації та технологій. Це дало змогу вплинути на час розробки, та якість продукту.

Розробка додатку відбувалася в середовищі JetBrains WebStorm.

2.1 Обрані методи реалізації

Для реалізації WEB-системи було прийняте рішення використовувати мову програмування JavaScript. Це дало змогу використовувати дану систему на таких платформах як: Linux, MacOS, Windows та інші.

2.1.1 React

React – це популярний веб-фреймворк, який розроблений командою із Facebook. Він в першу чергу був створений на розробку SPA .

React надає можливість:

1. двостороннє зв`язування;
2. маршрутизацію;
3. використання шаблонів;
4. динамічне завантаження модулів;

Це все дає нам можливість легко і швидко, зробити якісну WEB-систему.
За допомогою React , ми легко та зручно контрулюємо процесс маштабування системи. Можна для цього створювати моделі даних Immutable.js, RxJS.

Також у нас є можливість швидкого створення функцій за допомогою декларативних шаблонів. Розширення мови шаблонів вже залежить від вас і ваших підходів. Можна підчас розробки у IDE отримувати підказки, для більш зрозумілого і робочого коду.

Вся система складається із набору компонентів, які React рендерить під час користування системою. Кожний додаток на React, має хоча б один компонент.

В свою чергу компоненти користуються сервісами, які надають особливі функції, не пов’язані на пряму з представленнями. Це робить код модульним та ефективним. (рисунок 2.1)

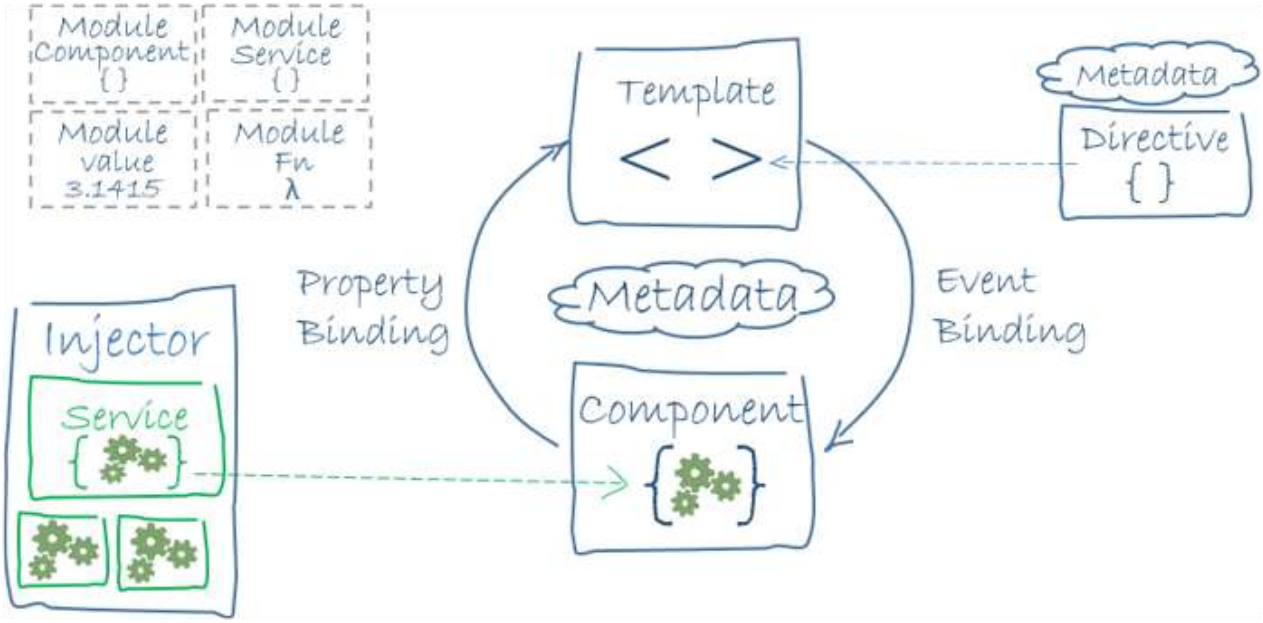


Рисунок 2.1 Схема архітектури React додатка

Для роботи з React використовується CLI, це так званий інтерфейс командного рядка React. За його допомогою використовуємо:

1. start – запуск проекта.
2. build – зборка проекта.

Щоб створювати шаблони, потрібно використовувати мову розмітки HTML та каскадні стилі CSS з препроцесором SASS.

2. 1.2 HTML

HTML – це мова гіпертекстової розмітки, вона є стандартною мовою для створення WEB сторінок.

HTML описує структуру WEB-сторінки семантично, тобто кожен блок має свою сутність.

HTML - елементи залишається засобом для створення структурованих документів, описуючи структурну семантику для тексту, наприклад абзаци, заголовки, списки, посилання, та інші елементи.

HTML5 – це основна версія стандарту HTML. Дана версія була створена, щоб покращити підтримку мультимедіа, зберігаючи його легко розуміючим людиною, та пристроями такими як синтаксичні аналізатори та браузери.

На сьогоднішній день HTML виконує таку роль, як:

- програвання відеозаписів;
- програвання аудіозаписів;
- малювання за допомогою Canvas;
- Drag’n’Drop.

2.1.2 CSS

CSS – використовується для опису презентації WEB-сторінок, включаючи кольори, шаблон і шрифти. За допомогою цього він дозволяє адаптувати презентацію до різних типів пристроїв, великі екрани, маленькі екрани або принтери. CSS не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML. Написання HTML окремо від CSS полегшує підтримку сайтів. Це так званий поділ структури на презентацію.

WEB-браузери використовують правила CSS, щоб впливати на вигляд елементів у браузері.

Правило CSS формується з:

1) набір властивостей, які мають значення для HTML - вмісту, наприклад, я хочу, щоб висота мого дочірнього елемента становила 50% від його батьківського елемента, а його шрифт відображався розміром у 1em;

2) селектори, придназначені для вибору елементів, які ви хочете змінити за допомогою стильових властивостей.

2.1.3 SCSS

SCSS — шаблонізатор, який предназначений для спрощення стоврення CSS-кода. Простіше кажучи, SCSS це мова, код якого спеціальною ruby-программою перетворюється в звичайний CSS код. Синтаксис цієї мови дуже гнучкий, і враховує багато деталів, які також потрібні в CSS. Більше того в ній є логіка така як (@if, each).

2.1.4 JavaScript

JavaScript – мова програмування. Без якої на сьогоднішній день важко уявити будь який WEB-додаток.

Також, JavaScript використовується у таких технологіях як:

- Front-end SPA – React, Angular.js, Angular;
- Back-end – Node.js;
- створення графіки – Pixi.js, Canvas;

JavaScript - має властивість об’єктно-орієнтовної мови програмування, але завдяки прототипності об’єктів вона відрізняється від традиційних мов ООП.

JavaScript містить певну кількість вбудованих в себе об’єктів: Boolean, Number, Math, Date, Global, Function, Object, Error, Array, String, RegExp. Окрім цього, також містить вбудовані функції, які не завжди є функціями або методами. Синтаксис JavaScript нагадує синтаксис C, але набагато простіший за нього, як описували розробники, для того, щоб зробити цю мову легкою для вивчення.

2.1.5 Redux

Redux - JavaScript бібліотека, яка була створена для керування станом програми. Дану бібліотеку можна використовував із різними фреймворками, але найбільше популярність, вона здобула у зв’язці із React.

Redux побудований на трьох концептах:

- **Єдине джерело істини (stage).** Він зберігає стан всього застосунку в дереві об’єктів. Одне дерево об’єктів дозволяє зберігати стан програми в процесі розробки і надає їй за рахунок цього прискорення;
- **Стан призначений тільки для читання.** Щоб змінити стан для цього є тільки один спосіб, це створити дію. Такий підхід гарантує, що ні перегляди, ні будь які виклики ніколи не зможуть змінити стан;
- **Зміни за допомогою чистої функції.** Редьюсери - (Reducer) чиста функція, яка приймає стан та дію, і повертає наступний стан. В процесі розробки дрібняться для зручності на ще менші редьюсери.

На рисунку 2.1.2 показано схему роботи Redux.

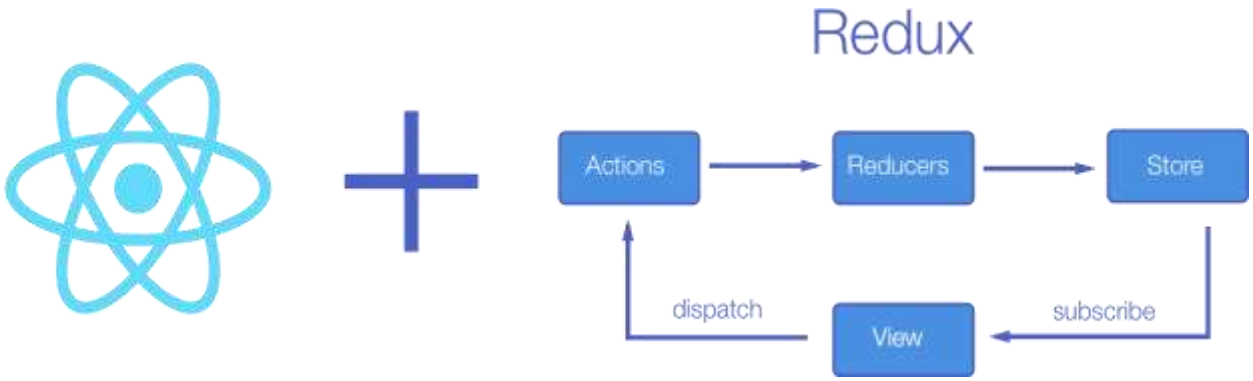


РИсунок 2.1.1 Схема роботи Redux

2.1.6 Серверна частина програмної системи

Серверна частина – та частина, що виконується на стороні серверу. Переважно, на серверній частина має більше бізнес-логіки. За допомогою неї робляться всі запити до бази даних.

2.1.6.1 Node.js

Node.js – це платформа з відкритим кодом для створення будь якого роду систем, написаних мовою JavaScript.

Node.js має такі властивості:

- асинхронна модель виконання запитів;
- система модулів CommonJS;
- JavaScript Chrome V8.

Для керування залежними модулями переважно використовується пакетний менеджер npm, але також можна і yarn.

NPM (Node Package Manager) - це менеджер пакетів для мови програмування JavaScript. Для середовища виконання Node.js по замовчуванню. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через веб-сайт npm.

Для обробки великої кількості запитів у Node.js використовує асинхронну модель запуску коду, яка базується на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів. Для з’єднання за допомогою мультиплексування використовується бібліотека libuv, а для створення пулу ниток (thread pool) використовує бібліотеку libeio, також потрібно згадатт, що виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Усі системні запроси, що створюють блокування, виконуються в пулі ниток, а після цього, як і обробники сигналів, передають дані результату своєї роботи назад через неіменовані канали (pipe).

Для збільшення функціональності на базі Node.js підготовлена колекція модулів,в якій можна знайти реалізацію SMTP, HTTP, XMPP, FTP, IMAP, DNS, POP3 серверів і клієнтів тарізних модулі для інтеграції з веб-фреймворками, обробники WebSocket і AJAX, конектори до СКБД (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори і систем авторизації (наприклад, OAuth).

По дефолту Node.js включає в себе набір різних модулів, у яких вже реалізовані типові операції такі як взаємодія з ОС, файловою системою, протоколами, мережею, утиліти для обробки даних.

2.1.6.2 **MongoDB**

MongoDB – це документо-орієнтовна система керування базами даних, яка не вимагає опису схеми таблиць. MongoDB займає положення між швидкими та масштабованими системами, які управляють даними у форматі ключ-значення, та реляційними СКБД.

MongoDB підтримує можливість зберігання документів в JSON- форматі, має гнучку можливість формування запитів, також може створювати унікальні індекси, які потрібні для різних збережених атрибутів, якісно та ефективно забезпечує зберігання, як малих так і великих бінарних об’єктів, підтримує журналювання операцій.

Основні можливості MongoDB:

- документо-орієнтовне сховище;
- гнучке формування запитів;
- можливість профілювання запитів;
- швидкі оновлення;
- зберігання бінарних даних різних великих обсягів;
- журналювання операцій;
- масштабування;
- підтримка індексів;
- асинхронна реплікація;

- набір реплік, шардінг;
- підтримка парадигми Map/Reduce.

MongoDB управляє наборами JSON- документів та інших, що зберагіються у форматі BSON. Пошук і зберігання в БД здійснюється за допомогою протоколу GridFS.

2.2 Висновки до розділу

Для створення програмного додатку у якості IDE було обрано JetBrains WebStorm, який всі зручності із усім, що пов’язано з написанням коду. Для розробки інтерфейсу було обрано мову JavaScript, фреймворк – React. На стороні сервера (Back-end) було обрано Node.js та базу даних –MongoDB. Такий стек дозволяє легко розробляти і підтримувати систему за необхідності. A SCSS та HTML стали засобами розмітки та стилізації.

Було обрано фреймворк React для створення WEB-додатку, який дає змогу швидко та зручно створювати складні за функціоналом додатки.

Із стеком технологій перелічених вище появилася можливість створити швидкий і зручний додаток, який буде складати на ринку конкуренцію своїм аналогам аналогам.

3. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

Розроблена система становить собою WEB-додаток, що має структуру представлену на рисунку 3.1

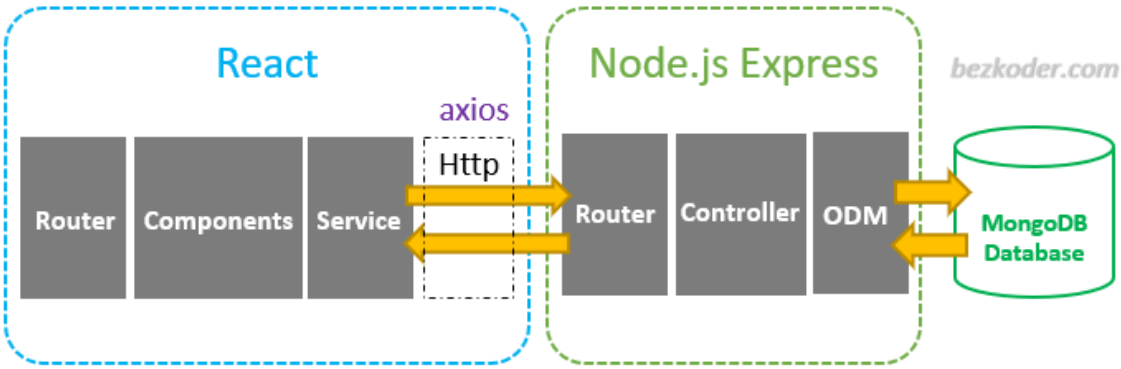


Рисунок 3.1 Архітектура програмної системи

3.1 Опис програмної реалізації програмної системи

WEB-додаток написаний за допомогою мови JavaScript та фреймворку React, що дає можливість створити швидко-зростаючий додаток, який складається із декількох модулів та модульної структури (малюнок. 3.2), а саме:

1. особистий кабінет адміністратора;
2. реєстраційної заяви абітурієнта;
3. перевірки рейтингового місця;
4. ядра додатку

3.2 Опис програмної реалізації серверної частини

Back-end реалізований за допомогою мови програмування JavaScript на платформі Node.js.

У якій описана основна бізнес-логіка, авторизації, сервіси для реєстрації, також робота з базою даних і моделі бази даних.

Архітектурним підходом було вибрано REST – RepresentativeStateTransfer.

3.2.1 REST

У самий REST дані попадають у невеликому обсязі таких форматів (HTML, XML, JSON). Будь-який REST- протокол змушений підтримувати кешування і не має залежати від мережевих прошарків також не повинен зберігати будь якої інформації про стан між парами “запит-відповіть”. За допомогою такого підходу легко масштабувати систему.

REST як і будь яка технологія має ряд архітектурних обмежень:

- **Відсутність стану**. Всі взаємодії які відбуваються між клієнтом і сервером не мають стану. Це означає, що кожен запит змушений вміщувати всю необхідну інформацію для обробки.
- **Кешування**. Системи, які написані з використанням REST змушенні підтримувати кешування. Це, означає, що дані які передаються з сервера, будуть містити дозвіл на кешування, та «час життя» кешування, якщо дозвіл відбувся.

3.2.2 Структура серверної частини

На рисунку 3.2.1 зображену структуру серверної частини системи

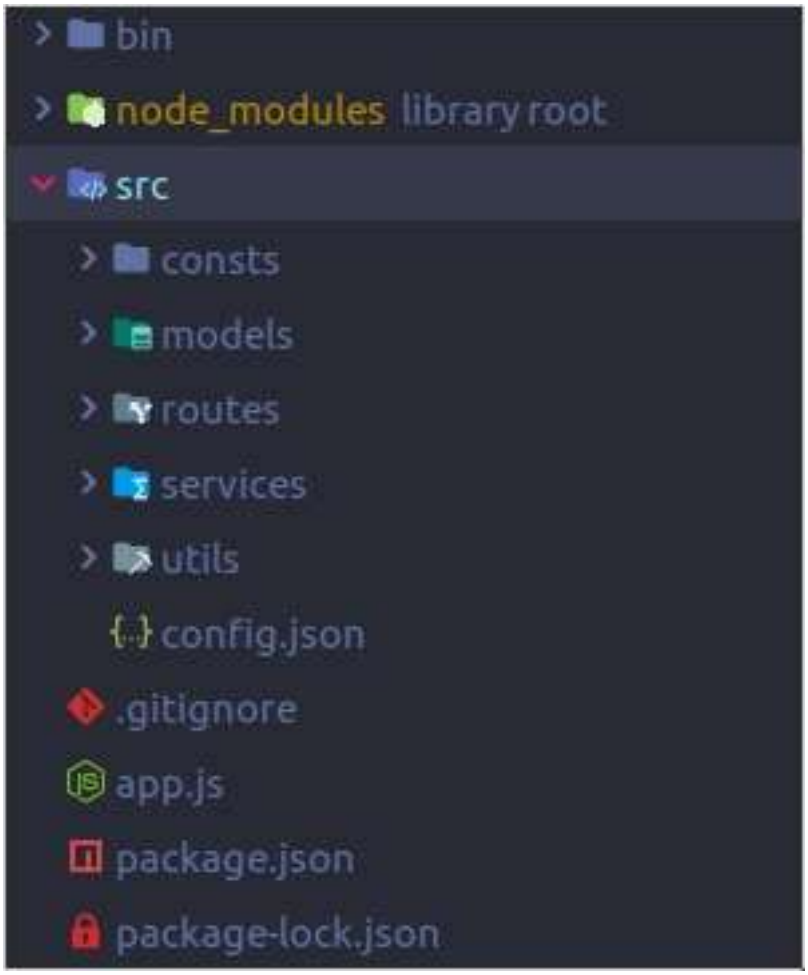


Рисунок 3.2.1 Структура серверної частини

- **node_modules –** директорія, де зберігаються всі бібліотеки, потрібні для роботи серверної частини;
- **src –** директорія з “самописним” кодом
- **src/consts –** директорія з константами;
- **src/models –** директорія з моделями бази даних;
- **src/routes –** директорія з файлами, що відповідають за обробку маршруту;
- **src/services –**сервіси;

- **src/utils** –засоби для не стандартних задач.
- **bin –** директорія з точкою входу ПЗ;

3.2.3 Сервіс взаємодії з базою даних

За взаємодію з mongoDb відповідає сервіс mongooseService.js. Він в свою чергу виконує всі функції асинхронно (async), за рахунок чого не блокує роботу сервера на час. В сервісі описаний такий функціонал як:

- **connect –** дає можливість підключення до бази даних;
- **disconnect –** зупиняє підключення до бази даних;
- **findDocuments –** здійснює пошук масиву записів в базі даних і повертає масив із записами, або пустий масив.
- **findOneDocument –** здійснює один пошук одного документу в базі даних і в залежності до умови документ або null;
- **createDocument –** здійснює створення нового запису в базі даних;
- **createModel –** здійснює створення моделі бази даних.

3.2.4 Сервіс реєстрації та авторизації

За реєстрацію і авторизацію відповідає сервіс, який називається usersService.js.

Коли відбувається реєстрації, в першу чергу перевіряється унікальність користувача за допомогою даних, які він надав ввівши у реєстраційній формі. Якщо користувача не існує із такими даними, то введені у даний момент дані, вносяться до бази даних і користувачу призначається відповідна роль та на клієнт відправляється сповіщення про те, що користувача успішно зареєстровано. А якщо користувач з введеними даними існує, то на клієнт відправляється сповіщення про те, що користувач з такими даними існує.

При авторизації, у першу чергу перевіряється наявність даних про користувача в базі даних, у якого має електронна адреса співпасти з введеною. Після чого відбувається перевірка пароля – в базі даних. Якщо і паролі співпадає, то користувач успішно авторизується. Якщо паролі не співпадають,

або якщо користувача з введеними даними не існує, на клієнт відправляється інформація про це.

3.3 Опис програмної реалізації клієнтської частини

Клієнтська (Front-end) частина WEB-системи написана за допомогою мови JavaScript у стеку їх фреймворком React, що дало змогу швидко створювати легко масштабуємий програмний додаток.

На рисунку 3.3.1 зображено структуру проекту

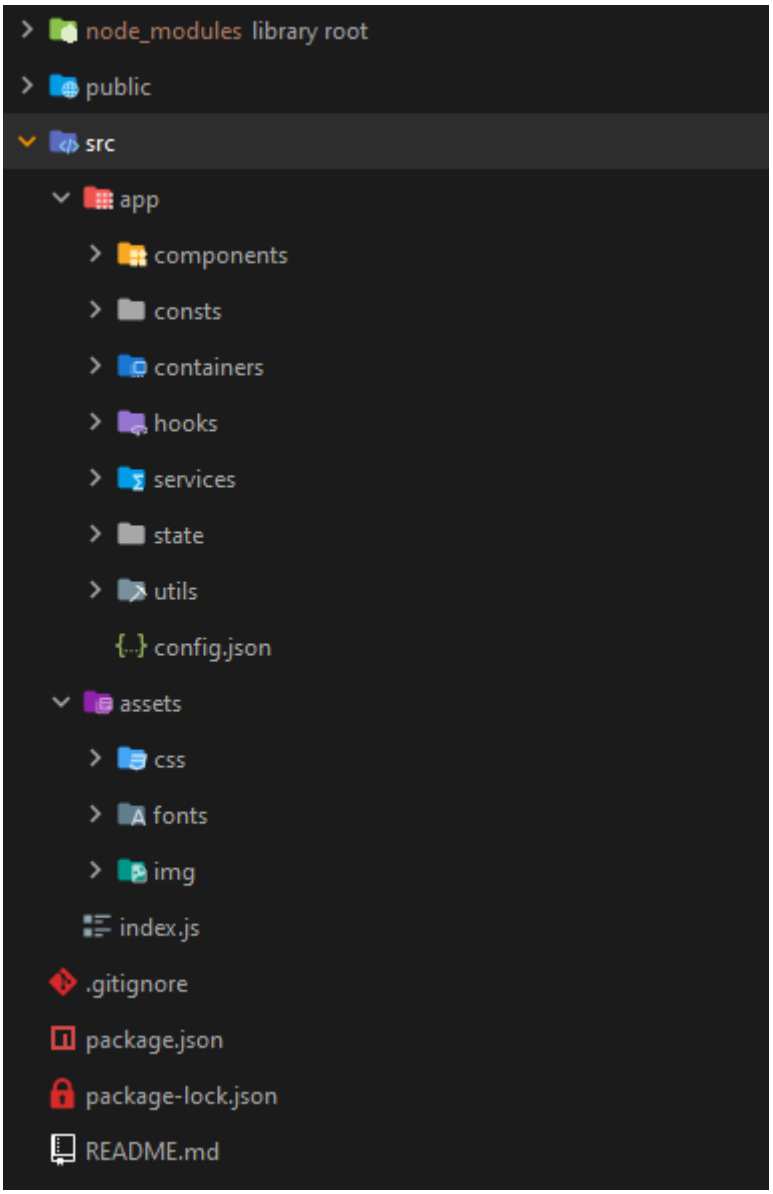


Рисунок 3.3.1 Структура проекту

- **node_modules –** директорія, де розміщуються бібліотеки, потрібні для коректної роботи додатку;
- **public –** директорія, де розміщуються статичні файли (index.html, favicon.ico, тощо);
- **src –** розміщуються з вихідними, файлами;
- **src/app –** директорія, де розміщуються файли додатку;
- **src/assets –** директорія, де розміщуються шрифти, стилі, зображення.

3.3.1 Реактивне програмування

Реактивне програмування – це таке програмування, як побудоване на потоках даних та розповсюдженні змін. А саме в умовах програмування має знаходитися можливість легко виразити статичні (static) чи динамічні потоки даних.

Реактивне програмування складається із на наступних концепціях:

- **Динаміка проти статики.** Реактивне програмування може набувати деяких станів чисто статичне (потоки встановлюються статично), або динамічним (потоки змінюються під час виконання програми).
- **Реактивне програмування .** Реактивне програмування вищого порядку називають реактивним програмуванням, якщо воно підтримує ідею про те, що потоки даних можуть бути використані для побудови інших потоків даних. Це може означати те, що результуюче значення з потоку є інший потік даних, який виконуються такою ж моделлю обчислення, що і перший.
- **Диференціація потоку даних.** В ідеальних випадках всі зміни даних поширюються миттєво, але це не завжди так на практиці. Як рішення, можна задати різні пріоритети для обчислення різних частин графу потоку даних. Це і називають диференційованим програмуванням.
- **Моделі обчислення у програмуванні.** Обчислення у програмах не завжди засновані обчисленнях у програмах, що будуються на стеках.

Окрім цього, коли будь які дані змінюються, то зміна поширюється на всі дані, які залежали від змінених даних.

- **Схожість з патерном «Спостерігач».** Програмування має велику подібність до патерну «Спостерігач», який використовується в ООП. Але, інтеграція концепцій потоку даних в мову програмування може збільшити ступінь деталізації графа.

3.3.2 Взаємодія з серверною частиною.

Для взаємодії коректної клієнтської частини з серверною частиною програмної системі, на боці клієнтської частини реалізовано сервіс із назвою (apiRequestService.js), який за допомогою AJAX типу GET або POST передає й отримує дані з серверної частини. Сервіс має наступний функціонал:

- **getAllUsers – GET –** получити дані про всіх користувачів;
- **registerUser – POST –** зареєструвати користувача;
- **auth – POST –** авторизувати користувача;
- **getUserDataById – POST –** получити інформацію про користувача за його ідентифікатором;
- **getAllGroups – GET –** получити дані про всі групи;
- **getAllFaculties – GET –** получити список всіх факультетів;
- **addNewGroup – POST –**створити нову групу;
- **getAllTeachers – GET –** получити список всіх викладачів;
- **registerTeacher – POST –**зареєструвати нового викладача.

На рисунку 3.3.2 зображено схему взаэмодії клієнта та сервера.

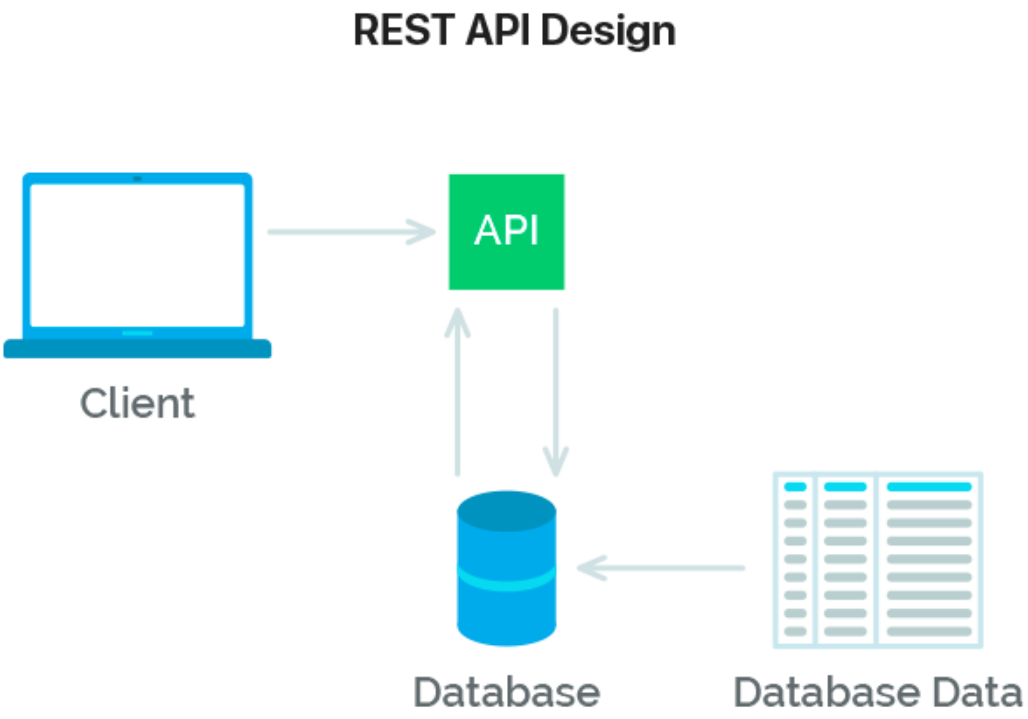


Рисунок 3.3.2 Взаємодія клієнта та сервера

3.3.3 Сторінка авторизації або реєстрації користувача

На рисунку 3.3.3 зображено структуру компоненту сторінки авторизації або реєстрації аспіранта.

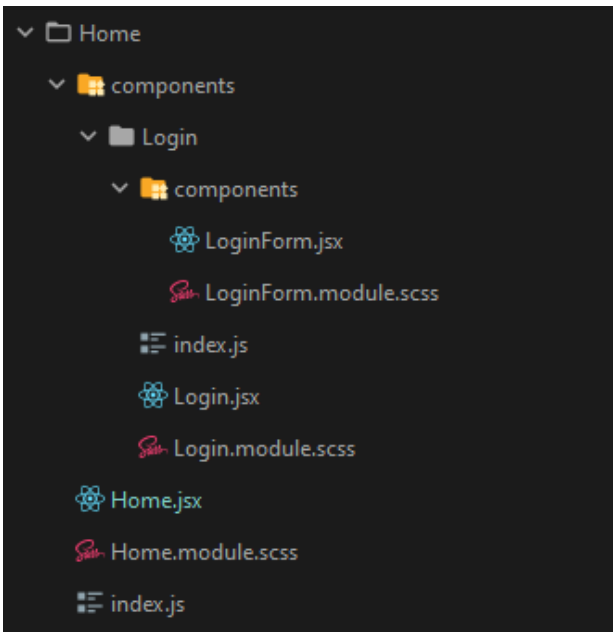


Рисунок 3.3.3 структура компонента сторінки авторизації чи реєстрації

Сторінка авторизації або реєстрації користувача, в основному, є головною сторінкою та в деяких моментах єдиною точкою входу користувача до будь якого програмного додатку. Із сторінки авторизації можна перейти на сторінку реєстраційної форми та сторінку рейтингового списку.

На сторінці авторизації чи реєстрації користувач має можливість здійснити авторизацію в системі для подальшого використання її функціоналу.

3.3.4 Сторінка подання реєстраційної форми

Сторінка подання реєстраційної форми призначена для подання вступником всієї необхідної інформації для його зарахування на третій рівень вищої освіти.

На рисунку 3.3.5 зображено структуру компоненту сторінки подання реєстраційної форми.

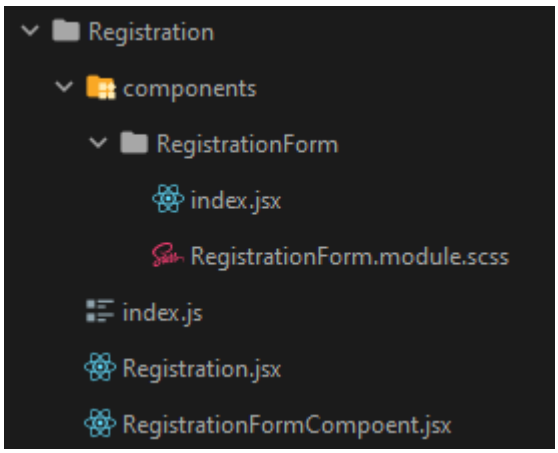


Рисунок 3.3.5 Структура сторінки подання реєстраційної форми

Реєстраційна форма заповнюється в три етапи (детальніше – в розділі 4 Інтерфейс користувача). Перейти до наступного етапу можна тільки після проходження валідації активним етапом. Реєстраційна форма потребує внесення наступних даних:

- прізвище;
- ім’я;
- по батькові;
- стать;
- дата народження;
- спеціальність – випадаючий список;
- факультет – випадаючий список;
- кафедра – випадаючий список;
- форма навчання – випадаючий список;
- форма оплати – випадаючий список;
- іноземна мова – випадаючий список;
- ВНЗ, що закінчив вступник;
- рік закінчення ВНЗ;

- освітній ступінь – випадаючий список;
- наявність відзнаки;
- електронна пошта;
- пароль;
- потреба в гуртожитку;
- середній бал диплому;
- кількість наукових публікацій;
- прізвище наукового керівника;
- додатковий коментар;

У момент, коли всі поля заповнені коректно, при відправці даних, валідація пропустить дані і дасть можливість відправити їх за допомогою POST-запиту на серверну частину програми.

3.3.5 Сторінка розкладу.

Сторінка розкладу призначена для перегляду вступником всієї необхідної інформації по розкладу для його зручного і ефективного навчання.

На рисунку 3.3.6 зображено структуру компоненти розкладу.

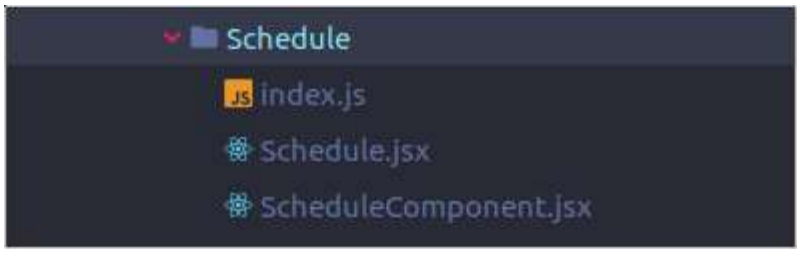


Рисунок 3.3.6 Структура сторінки розкладу

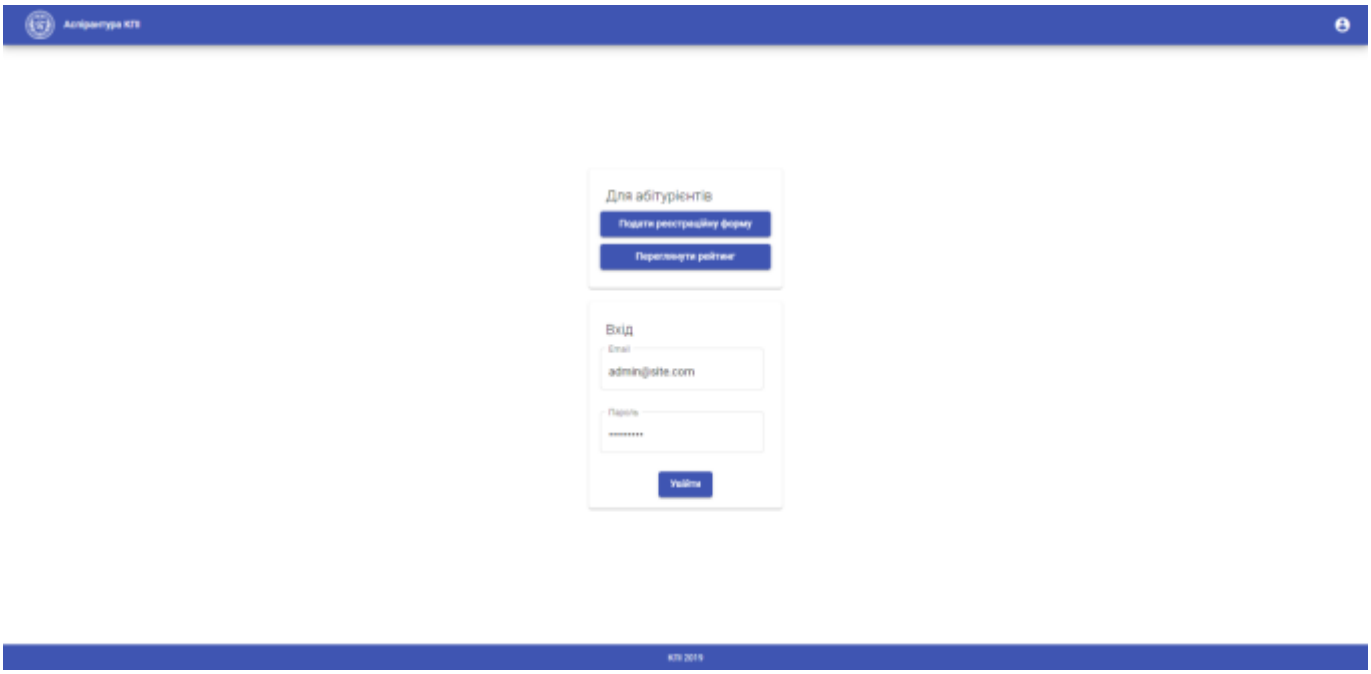
4 ІНТЕРФЕЙС ВЗАЄМОДІЇ ТА РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Розроблений WEB-додаток є доступний для всіх бажаючих, хто має вихід в інтернет і установлений браузер на своєму пристрої.

4.1 Інтерфейс системи

На головній сторінці додатку (рисунку 4.1) користувач із роллю абітурієнта може подати реєстраційну форму або переглянути рейтинги.

Також адміністратор може авторизуватися в системі, щоб отримати можливість працювати із даними вступників.



Малюнок 4.1 Головна сторінка системи

4.1.1 Функціональні можливості абітурієнта

Реєстраційна форма складається з трьох етапів.

На першому етапі користувач із роллю абітурієнта вводить свої дані (рисунок 4.3):

1. прізвище, ім’я по батькові;
2. стать;
3. прізвище;
4. ім’я;
5. по батькові;
6. стать;
7. дата народження.

Рисунок 4.3 Перший етап подачі реєстраційної форми.

На другому етапі користувач із роллю абітурієнт, потрібно вибрати спеціальність, факультет та кафедру. (рисунок 4.5)

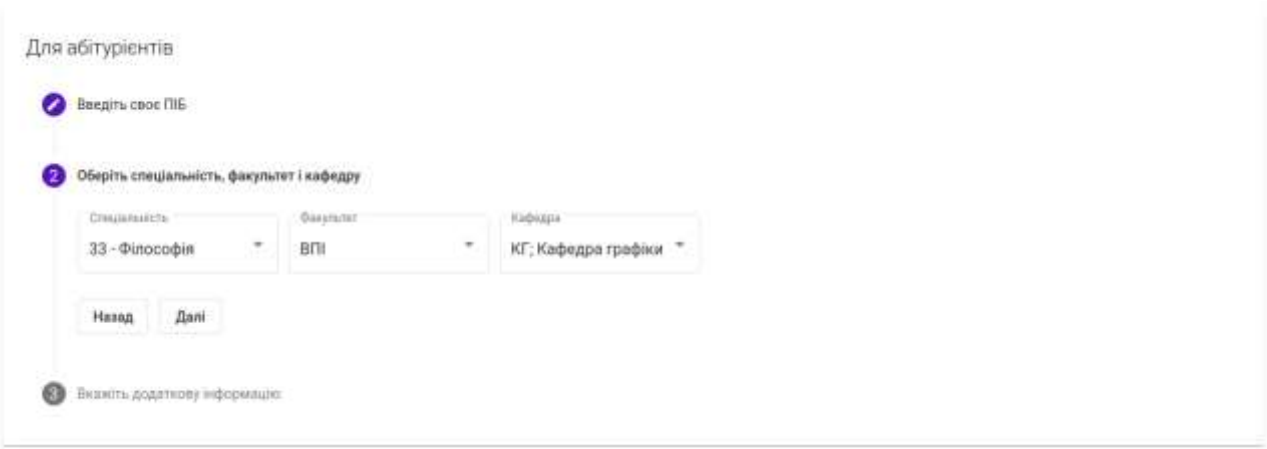


Рисунок 4.5 Другий етап подачі даних для реєстрації

Третім етапом подачі є заповнення даних полів (Рисунок .4.6):

1. форма навчання;
2. форма оплати;
3. іноземна мова;
4. ВНЗ який закінчив абітурієнт;
5. рік випуску;
6. освітній ступінь;
7. диплом з відзнакою;
8. електронна пошта;
9. телефон;
10. чи потребує місця в гуртожитку;
11. середній бал диплому за 100-бальною шкалою;
12. кількість публікацій;
13. прізвище наукового курівника;
14. особливі відзнаки;
15. додаткова інформація;

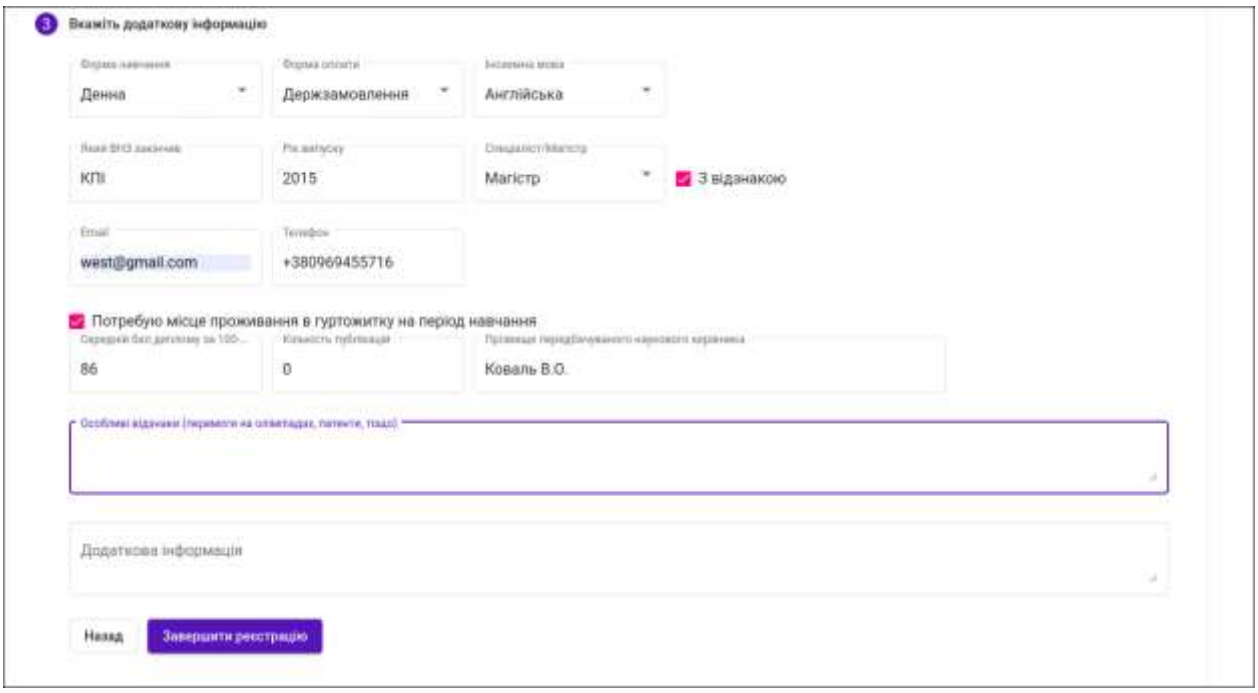


Рисунок 4.6 Третій етап подачі даних для реєстрації

Після входу у свій особистий кабінет, абітурієнт може переглядати свій рейтинговий бал. (Рисунок 4.7)

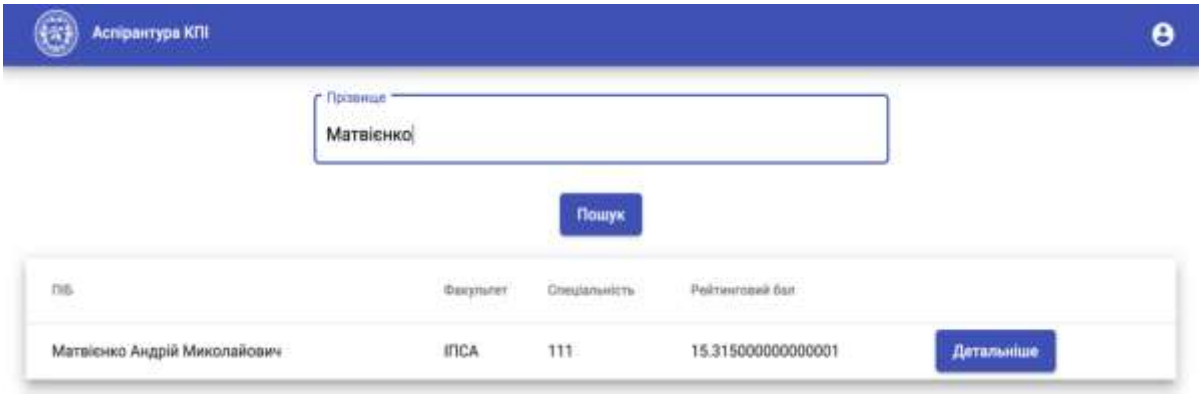


Рисунок 4.7 Форма перевірки рейтингового балу

Корисутвач із правами абітурієнта має змогу переглянути список вступників та визначити своє місце в рейтингу (Рисунок 4.8)

Рисунок 4.8 Список рейтингу вступників

4.1.2 Функціональні можливості адміністратора

Після того, як відбулася успішно авторизація користувач із правами адміністратора, може переглянути список поданих заяв (Рисунок 4.9)

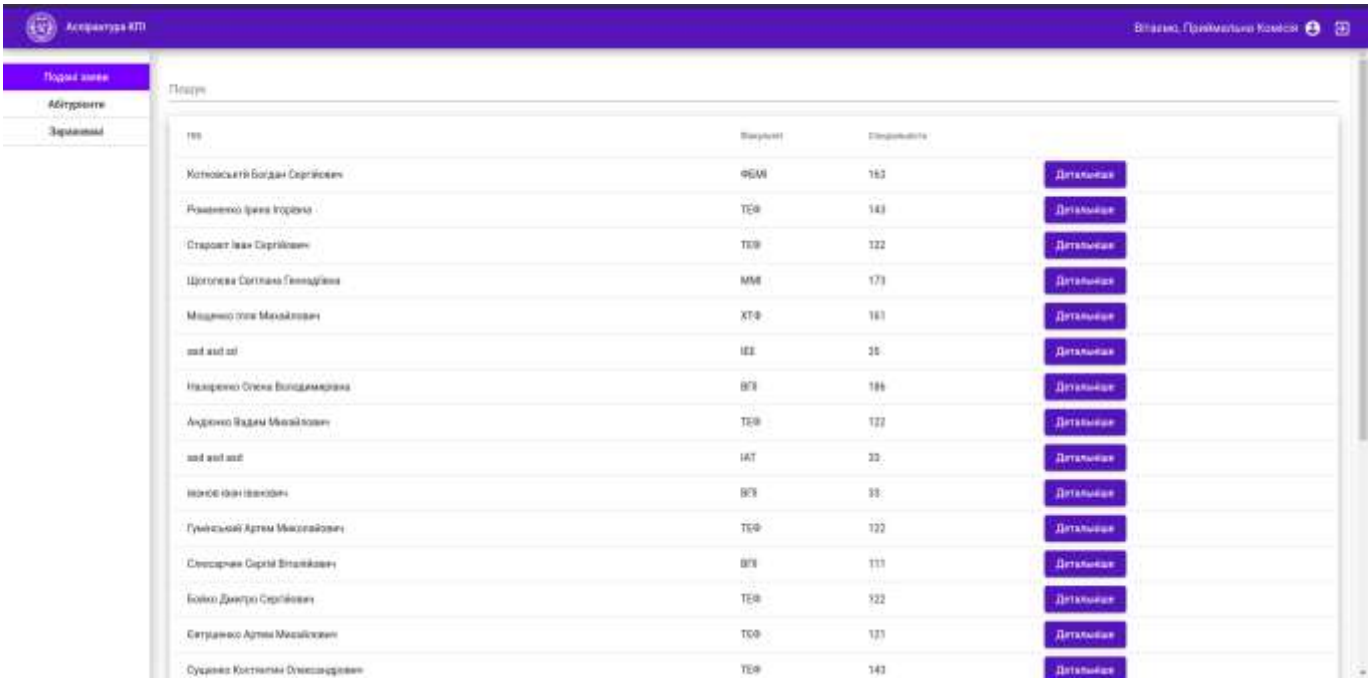


Рисунок 4.9 Список поданих заяв

На детальній сторінці (рисунок 4.10), користувач із правами адміністратора може переглянути дані конкретного вступника та прийняти рішення.

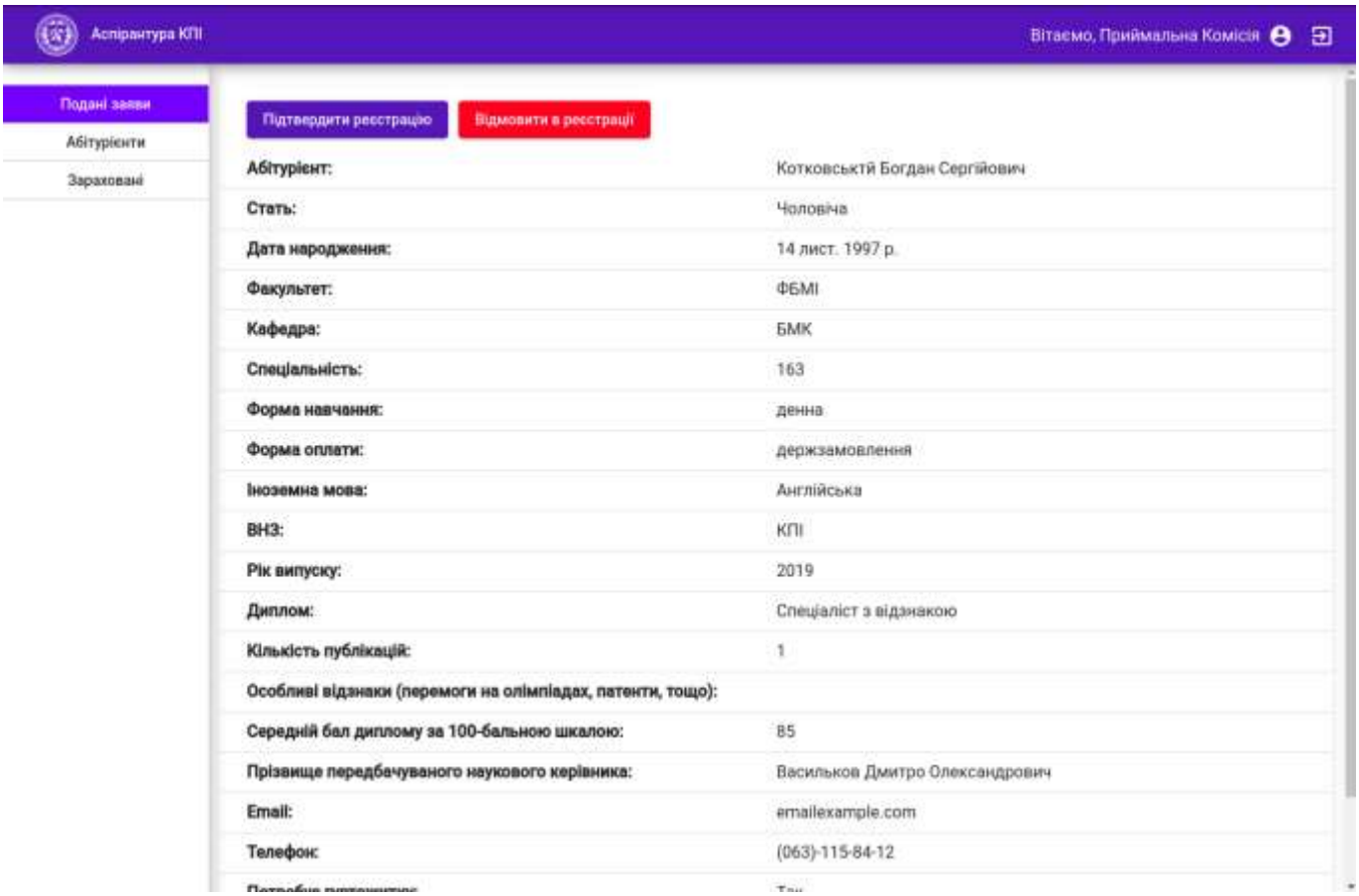


Рисунок 4.10. Детальна сторінка заяви вступника.

У пункті меню на сторінці абітурієнтів, користувачу із правами адміністратора доступна таблиця із вступниками, яких реєстрація була підтверджена. (малюнок 4.11)

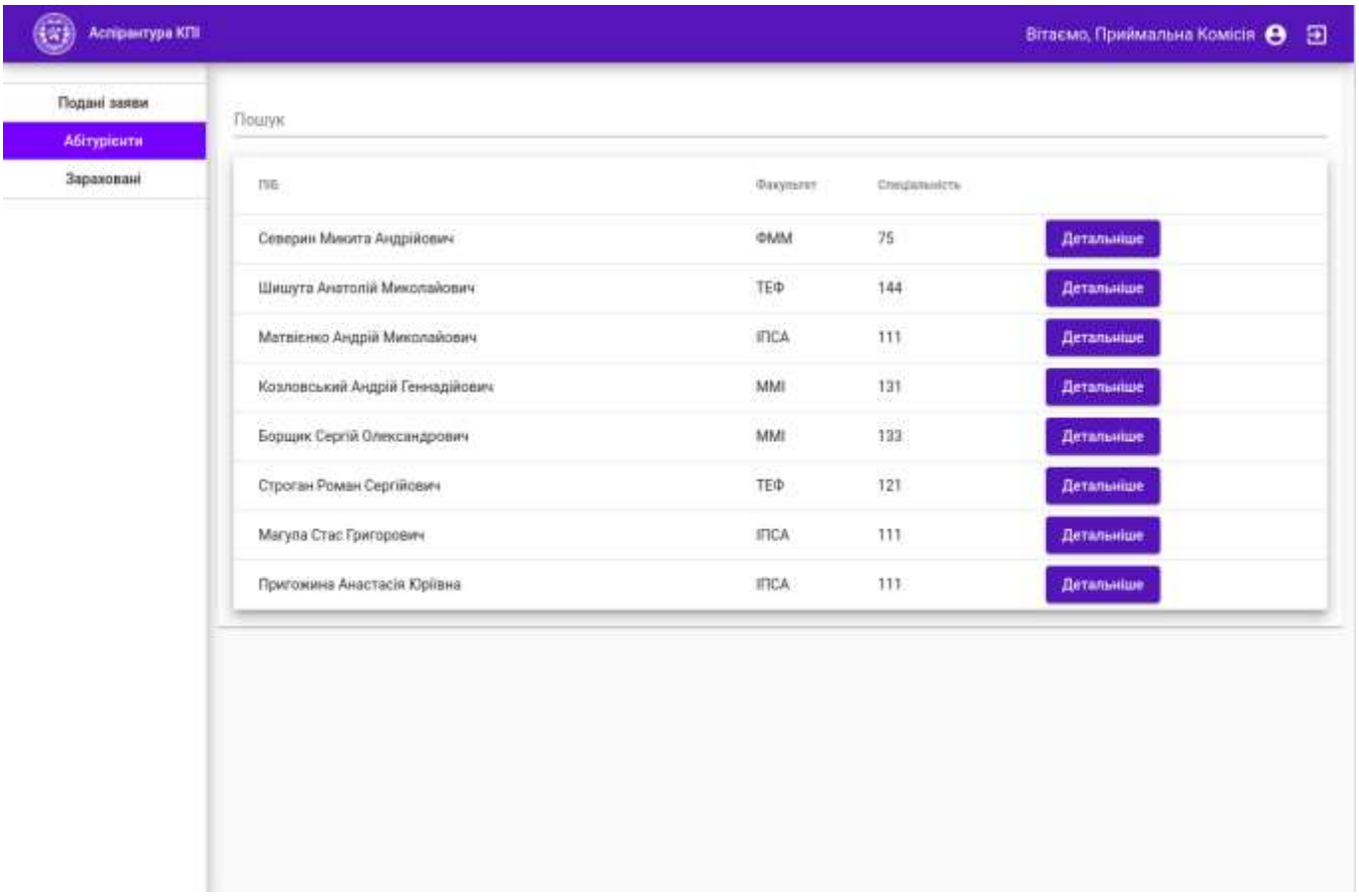


Рисунок 4.11 Список абітурієнтів, реєстрацію яким було підтверджено.

На детальній сторінці вступника (рисунок 4.12), користувач із правами адміністратора може переглянути дані абітурієнта, а також внести бали за вступні екзамени до системи. Система згенерує рейтинговий бал і внесе зміни в рейтинговий список.

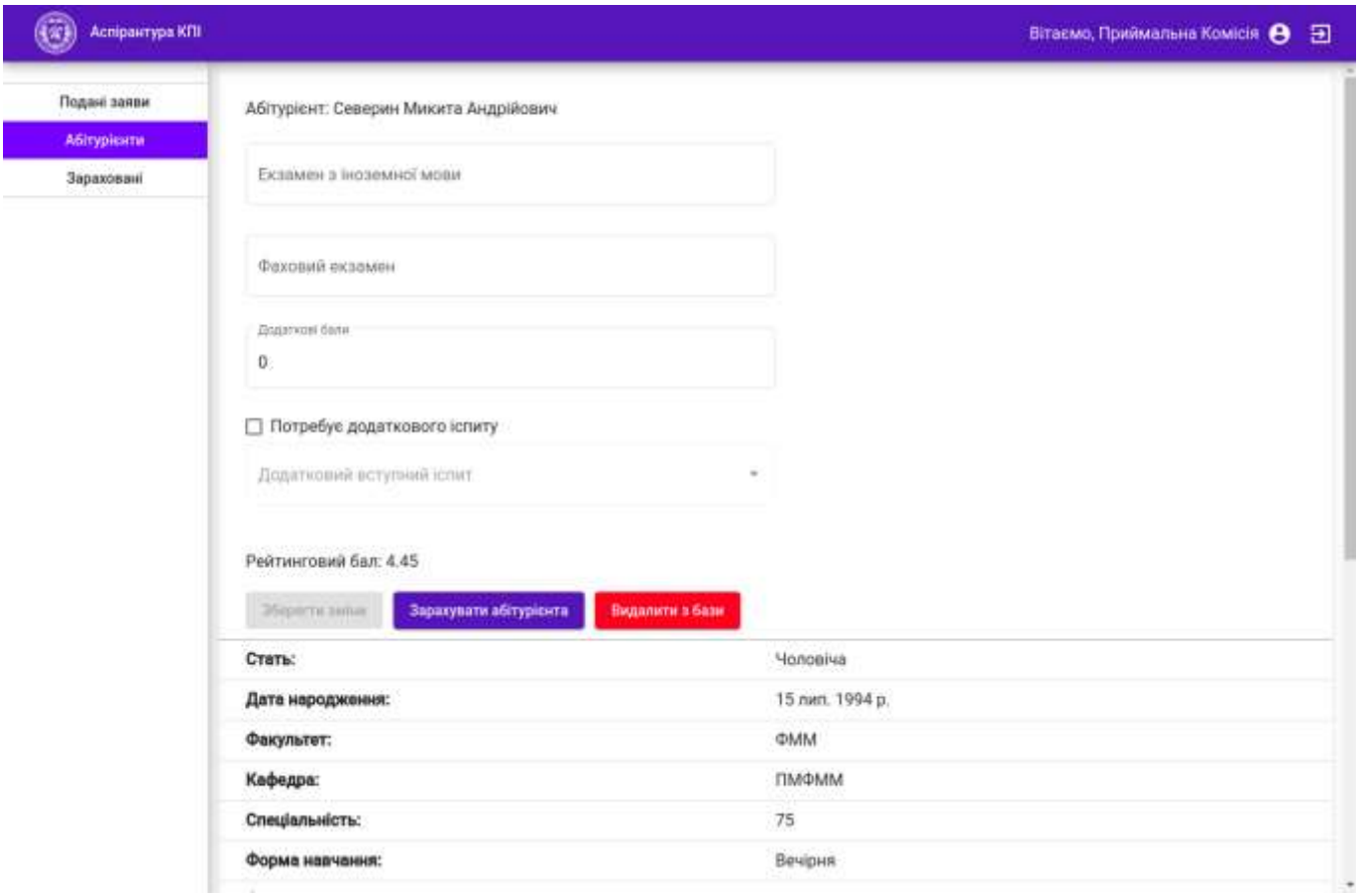


Рисунок 4.12 Детальна сторінка абітурієнта.

На сторінці “Зараховані” відображаються зараховані абітурієнти (малюнок 4.13), користувач із правами адміністратора має змогу вносити всупників до груп (малюнок 4.14).

Рисунок 4.13. Таблиця зарахованих вступників.

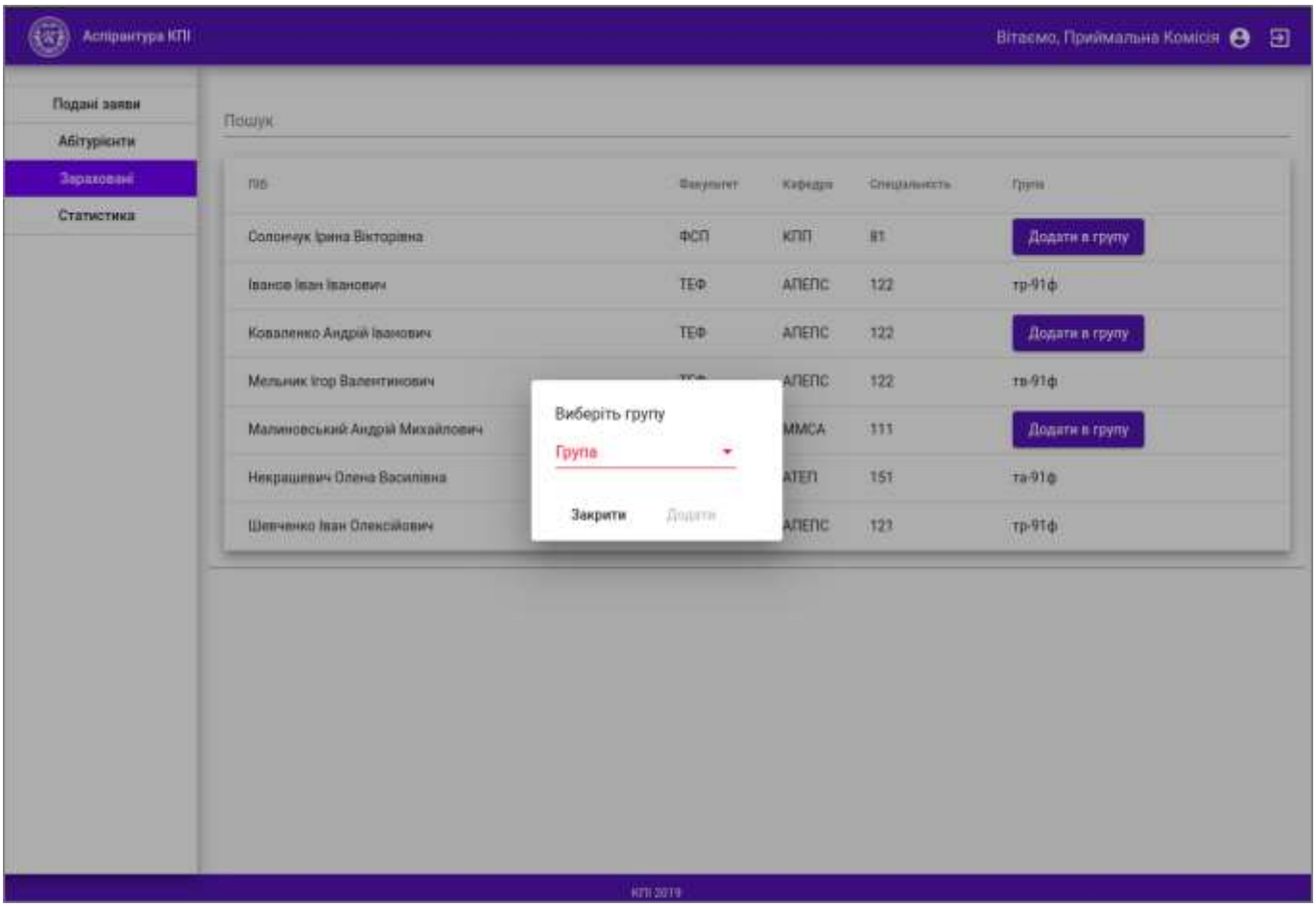


Рисунок 4.14. Додавання абітурієнта до групи

4.1.3 Функціольні можливіості аспіранта

Після того, як абітурієнт успішно зарахований на навчання, він стає аспірантом і внього появляється настурний функціонал, мій профіль (Рисунок 4.15).

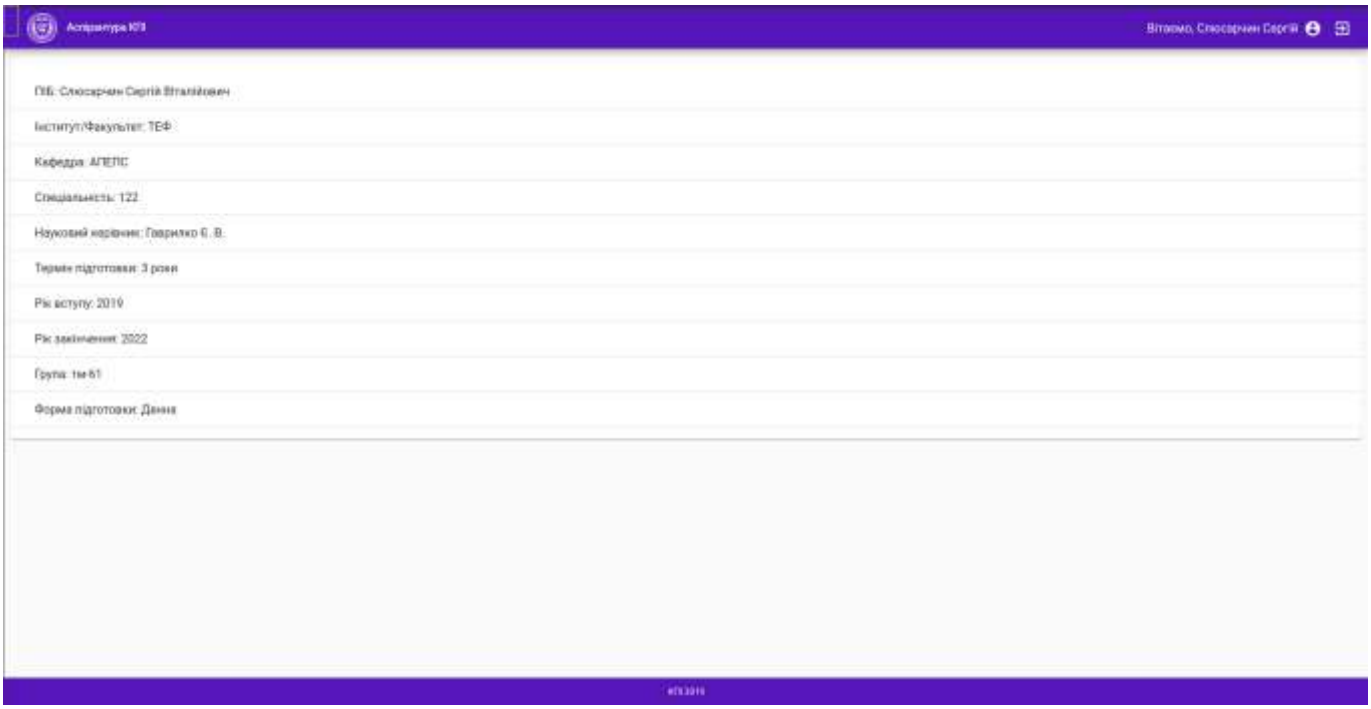


Рисунок 4.15. Перегляд особистого кабінету

Також аспірант може переглянути індивідуальний план. (Рисунок 4.16)

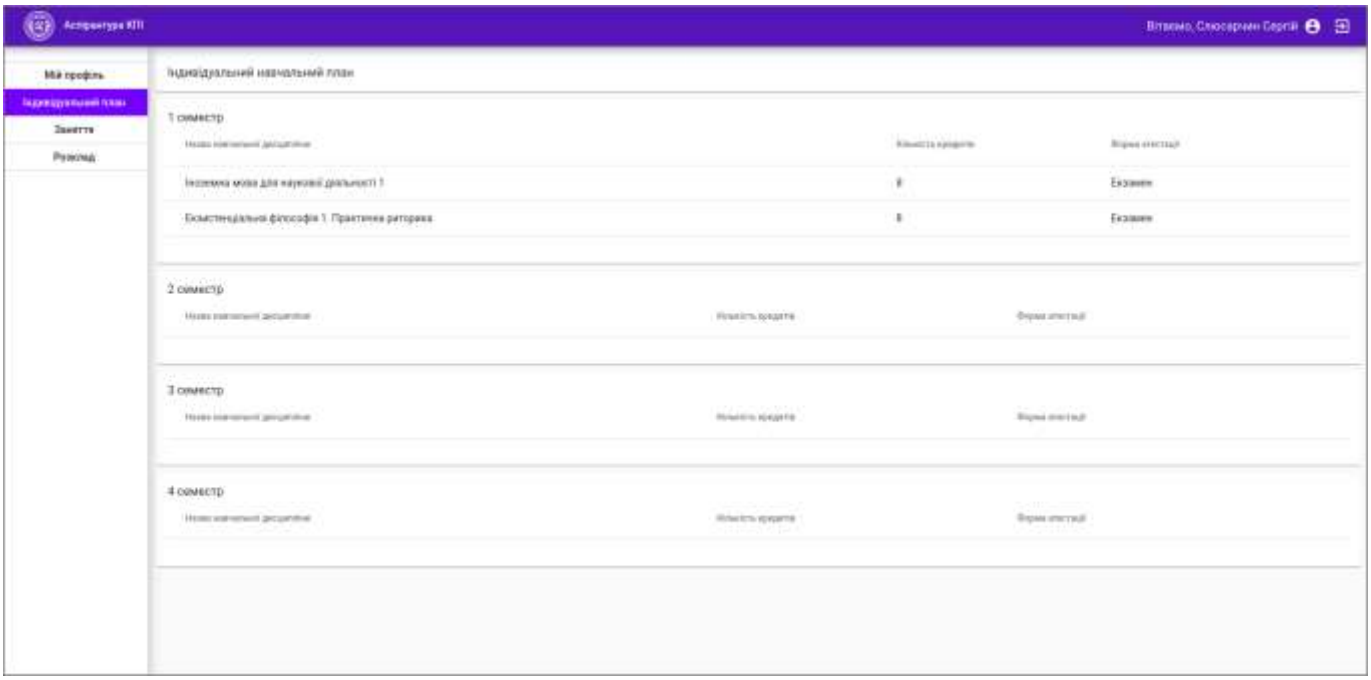


Рисунок 4.16. Перегляд Індивідуального плану

У пункті меню можна перейти на тсорінку “Заняття” (Рисунок 4.17). Де аспірант може переглянути свою успішність із предметів.

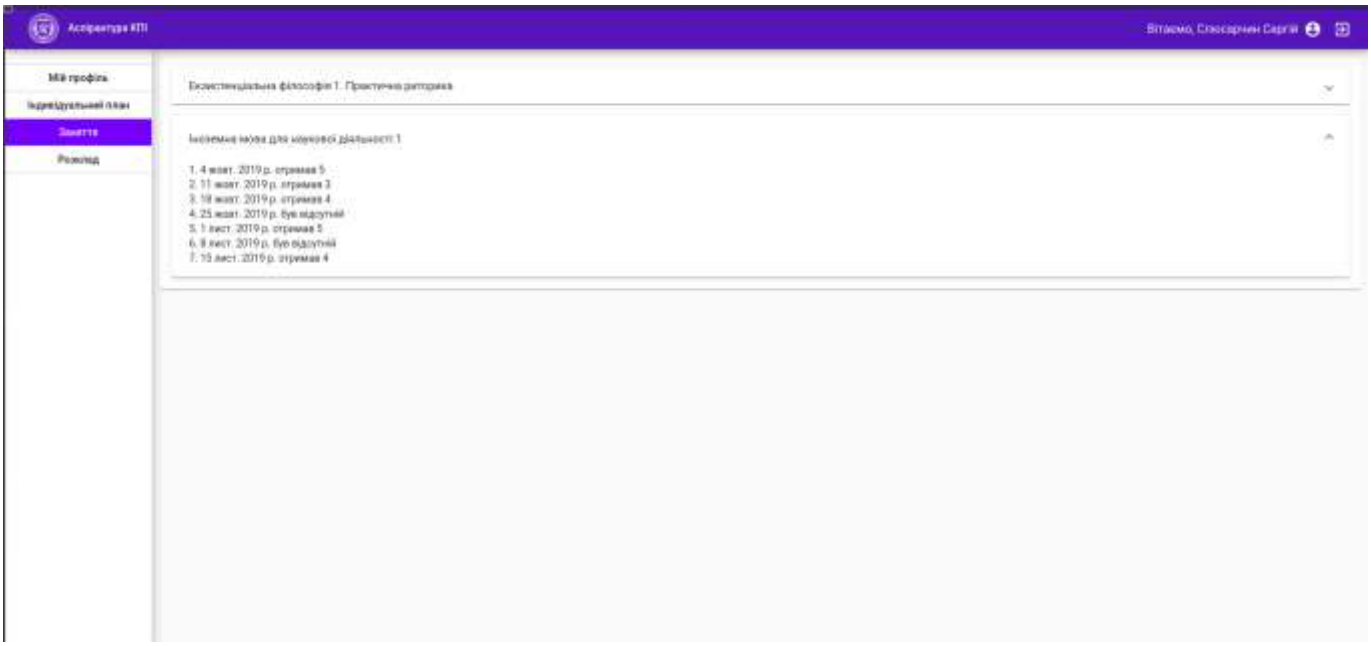


Рисунок 4.17. Перегляд заннять.

Висновки до розділу 4.2

У даному розділі коротко описується принцип роботи із програю “Навчання”, що дає змогу проводити набагато зручніший процес навчання чим це було до цього у вищих навчальних закладах.

WEB-додаток дозволяє абітурієнту подати документи на вступ через реєстраційну форму, після успішного прийняття на навчання із правами аспіранта. Користувач получає можливість переглядати особисті дані, індивідуальний план, заняття та інше.

А користувач із правами адміністратора системи має доступ до функціоналу, де може з легкістю створювати навчальні групи, розприділяти аспірантів по групам та створювати нових викладачів та інше.

Користувач із правами викладача може перегялдати особисту інформацію, оцінювати аспіратів та відмічати їхнє відвідування занять.

Підчас тестування додатку не було знайдено нічого, щоб погіршувало роботу системи, тому додаток можна починати використовувати і сьогодні.

ВИСНОВКИ

Підчас дослідження був розроблений додаток “Навчання” для системи підготовки, навчання науково – педагогічних кадрів вищої кваліфікації в аспірантурі. Даний додаток був реалізований у вигляді WEB-застосунку. Варто відмітити, що були отримані навички роботи з material-ui та redux-form, які містять в собі великий обсяг розроблених компонентів та функціоналу.
Із допомогою використання мови програмування JavaScript та фреймворку React на стороні WEB-застосунку і платформи Node.js із сторони серверної частини, вдалося отримати поставлені цілі та забезпечити підтримку системи та можливість розширюватись.

Використання розробленого додатку, автоматично забезпечить велику кількість людей для взаємодії з продуктом, так , як додаток працює на всіх ОС, все що потрібно це браузер та інтернет. Таким чином, розроблена система спростила роботу адміністраторам і в свою чергу абітурієнтам навчальних закладів. Також гарантувала забезпечення прозорого процесу вступної компанії.

В Адмініструванні, також получилося спростити й пришвидшити всі процеси, які раніше були паперовими і часозатратними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Міністерство освіти і науки України — Київ, 2020 – 53 с.
2. Флэнаган Дэвид «JavaScript. Подробное руководство» [Електронний ресурс].
3. Michael McMillan «Data Structures and Algorithms with JavaScript» [Електронний ресурс].
4. Руководство по https://material-ui.com/ [Електронний ресурс].
5. Руководство по HTML http://htmlbook.ru/html [Електронний ресурс].
6. Руководство по HTML и CSS. Проектирование и дизайн веб-сайтов [Електронний ресурс].
7. Руководство по React https://learn-reactjs.ru/home [Електронний ресурс].
8. Руководство по Redux and React https://litportal.ru/avtory/a-benks-piter/kniga-react-i-redux-funkcionalnaya-veb-razrabotka-786130.html [Електронний ресурс].
9. Руководство по mongoDb https://dmkpress.com/files/PDF/978-5-94074-831-1.pdf [Електронний ресурс].